IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, May 9-10, 1991

A STUDY OF A HIGH BANDWIDTH AND LOW LATENCY INTERCONNECTION NETWORK IN PIE64

Eiichi TAKAHASHI, Hanpei KOIKE, and Hidehiko TANAKA
{eiichi,koike,tanaka}@mtl.t.u-tokyo.ac.jp

Tanaka Laboratory, Department of Electrical Engineering,
Faculty of Engineering, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113, JAPAN

Abstract: PIE64[1] is a parallel inference machine. The goal is fast execution of large scale knowledge processing. Generally speaking, an interconnection network(IN) is one of the key to designing a parallel machine and affects a total system architecture. We designed the IN[2] of PIE64 with the aim of maximizing its performance.

In this paper, we first discuss the IN suitable for PIE64, and propose an IN, which has the following features: circuit switching, non buffering, multistage, dynamic load balancing support, and duplicated network. Next we consider its hardware implementation, and report the assembling process. Finally we show and discuss its electrical characteristics.

Introduction

PIE64 is a parallel knowledge processing machine that aims at accelerating symbolic processing. One of main features of PIE64 is that two homogeneous IN connect 64 processing elements(PEs)[4]. Figure 1 shows the block diagram of the PIE64, and figure 2 shows the block diagram of one of the two INs. An IN is one of the key to designing parallel architecture, because INs performance and characteristics directly affect the entire performance of the system.

The IN of PIE64 has three major features: circuit switching, multistage network, and dynamic load balancing. They provide each PE with channels that has sufficient bandwidth to communicate with other.

Each IN is multistage network with 64 inputs and 64 outputs, and consists of 48 switch nodes that are 4 inputs × 4 outputs, 32-bit width crossbars. Because multistage networks have more channels than other topologies, it is difficult to implement them physically. PIE64 has solved this problem by some ideas of assembling the hardware[5].

At present, we have finished testing the IN of PIE64, and have been making processing units. A dedicated hardware debugger, TAKO, enabled the following tests with condition similar to operating actually: to confirm its function, to debug, to test its maintenance part, and to measure its electrical characteristics and its heat radiation state.

The paper first examines the structure of the IN of PIE64. Next, it describes the implementation of IN hardware and reports its assembling process. Finally, it discusses its electrical characteristics.

Interconnection Network of PIE64

PIE64 runs programs written in Committed Choice Language. Such programs produce many small processes dynamically, and the PE is assigned and processed each of such processes as a unit. This indicates that most of data transferred through the IN are one or two words long, that such short data are frequently transferred, and that there are many fine-grained bidirectional data transfers, for example, PE sends memory address to other PE and waits for the read. Parallel machines with these communication characteristics, including the PIE64, need low latency and high bandwidth IN.

In addition, since parallel machines with more processors tend to be harder to be implemented physically because of the increasing rate of IN hardware,INs should have a simple structure as much as possible for experimental hardware. Therefore, we proposed an IN of PIE64 that meets these requirements. It has the following features: circuit switching, multistage network, non-buffering, duplicated network, and distributed routing. First, we will examine the first two features.

Circuit Switching: PIE64 adopted not "packet switching" but "circuit switching" based on the following examinations.

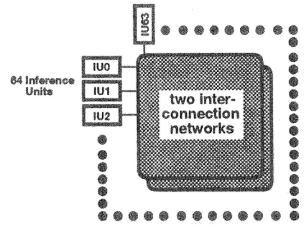

Figure 1: Architecture of a parallel inference machine PIE64

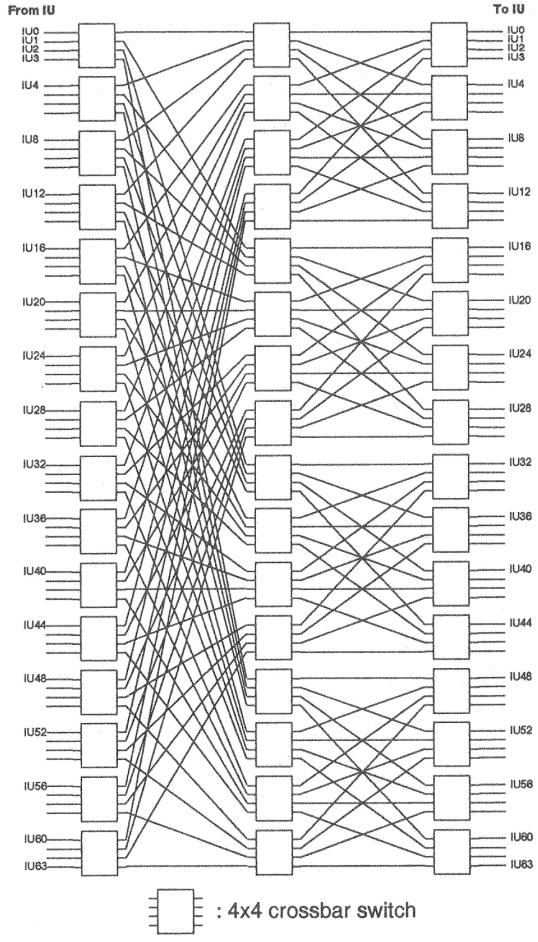

Figure 2: IN topology of PIE64

- Because data and routing information solely flow through such INs, there are no additional informations that yield any overhead.
- Because the INs provide bidirectional channels once establishing connections, they have the advantages in case of necessity for short turnaround, for example, PE waits for remote data after sending its address.
- Though the INs have some overhead for establishment of connection, they have the advantages of little transfer cost.
- In use of the same device technology, it is expected that the INs are smaller in physical size and faster in clock speed.
- It costs less to design, to make, and to debug.
- The IN has primitive functions and PE utilizes them in various manner, therefore PEs can maximize the performance.

On the other hand, though some problems are pointed out, one can cope with them as follows:

- *It is necessary to deal with the blocking in the INs without performance degradation*
  Not IN but PE deals with timeout of connection requests due to the blocking. This treatment enables the IN hardware simpler, and the PE more flexible to deal with the blocking.
- *In the blocking network, heavy traffic results in the saturation*
  The frequency of the blocking can be decrease using duplicated IN. In addition, the use of both dynamic and static load balancing strategies lowers the upper limit of traffic.

Multistage Network: Next, we compare multistage network, which we adopted, with other topologies such as hypercube, mesh, and crossbar.

- *All distances between each processors are equal*
  Such INs are flexible to a wide range of programs. However, in certain conditions they are not always the most suitable because they cannot keep any static connection patterns that are the most suitable for executing some kinds of programs. Using the INs, the overhead of dynamic load balancing is little because of a simplified model for load balancing; in contrast, other kinds of topologies such as hypercube and mesh cannot always utilize the dynamic load balancing on account of much overhead.
- *The diameter of such INs is rather small*
  The diameters[6] of mesh, hypercube, multistage, and crossbar are calculated as follows:

  k-dimensional mesh $\ldots O(\sqrt[k]{n})$ $\quad (k \leq 3)$
  Hypercube $\ldots\ldots\ldots O(\log_2 n)$
  Multistage $\ldots\ldots\ldots O(\log_k n)$ $\quad (k = 2, 4, 8, \ldots)$
  Crossbar $\ldots\ldots\ldots O(1)$

  This shows that the diameter of multistage network is rather short(the base of logarithm k express the number of inputs or outputs of switch nodes). In comparison with hypercube, whereas the diameters are equal the multistage network needs more hardware since the multistage network topology consists of more arcs and more nodes than hypercube topology. Crossbar needs much more hardware.
- *The INs are expandable into larger INs or INs of more bit width*
  Mesh, hypercube, and multistage network are expandable into larger networks with the same hardware.

Other features: This section examines the other features of the IN.

- *Non-buffering in an switch nodes*
  In circuit switching network, if the transfer time delay of IN is shorter than cycle time of PE, transfer data do not have to buffer in switch node on the way. Therefore, the latency of such INs can be low. In order to reduce the transfer delay, each hardware component in the system should be compact.
- *Dynamic load balancing support* (Fig. 3)[3]
  Switch nodes that consist of the IN are initially in a state of no connections. The IN involves comparator, with which it compares each PE's load information that is given to the switch through its unused output ports in the reverse direction. The IN memorizes the PE with the lowest load and delivers the value through its input ports to all the PE that do not use its communication channel. Based on this mechanism, the IN connects to the lowest load PE as soon as such a request is made. This mechanism is simple, easy to implement, and effective, as well as not yielding any overhead, because it utilizes unused resources. However, Expressions of the load information is the key to utilizing this mechanism.

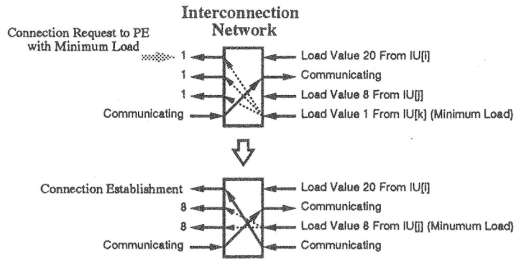

Figure 3: The way how the IN supports to balance loads dynamically

- *Expandability*
  The IN can expand its bit width of channels into multiples of 8 bits, while it can expand the number of ports into 256 with four stages.
- *Duplicated IN*
  Because the IN of PIE64 consists of two independent networks of the same structure, the blocking factor much decreases. As other merits, using two independent comparators in the IN enables processors to express their load information in the form of pairs.

The Implementation

The previous section considered the IN and examined its properties.

This section describes the following problems in implementing the IN: basic structures, placement of PE boards, implementation of the IN, assignment of switch nodes, and assembly.

Basic Structures: First of all, we determined the following two basic plans:

- The number of processors is 64, because we think that it is more important to construct parallel machines with joining some amount of PEs of high performance than many PEs of low performance.
- The IN consists of two independent networks so as to decrease the blocking factor and to increase the throughput.

Next, in order to construct the multistage circuit switching network, we developed four inputs and four outputs, eight bit width crossbar LSI (Switching Unit, SU). It has:

- A distributed router that can be operable with which the multistage network is constructed
- Six communication control lines besides eight data lines
- A comparator that supports dynamic load balancing
- Scan paths for the internal diagnosis

Since the PE has 32-bit internal busses, the bit width of the switch node was determined to be 32. The switch nodes were constructed of four SUs,

Arrangement of PE Boards: The IN is indirect network[6]; it is logically located in the center and surrounded with PEs. Therefore, it is reasonable that the IN and the PEs are placed thus in the same construction physically. In order to realize this construction, we prepared original racks showed in fig. 4.

The Implementation: IN boards, which are placed in the center, are shaped into a cube, which is divided just into two parts like fig. 4.

At first we planned to connect IN boards by means of connectors that connect them directly. However, we rejected this because of the following reasons:

- Ordinary connectors connected between the boards are useless because too many lines need connected.
- If high density connectors are used, it is difficult to connect pins of the connector with PCB(Printed Circuit Board).

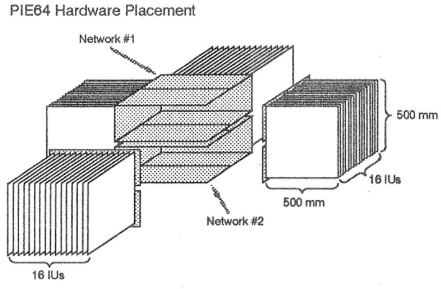

Figure 4: PCBs' arrangement of PIE64 hardware

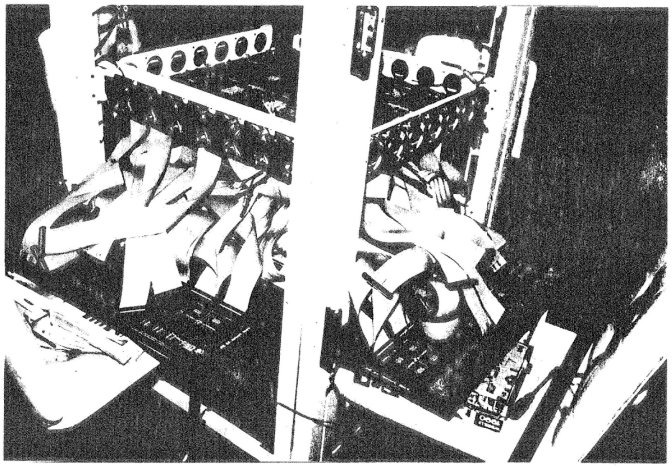

Figure 5: The IN hardware(under assembly)

- It is hard to test the switch nodes separately.

In addition to these issues, the PCB of high density wiring have the following problems:

- Some problems such as noise induction and crosstalk are critical.
- It is difficult to make delay of 32 wires (one word) uniform.

Therefore, we determined to connect the switch nodes by means of FRCs (Flat Ribbon Cables). This approach has several advantages as follows:

- Its data transfer characteristics is stable.
- It can make all delays of a set of wires uniform.
- It enables each switch node to be tested separately.

On the other hand, though it tends to be misconnected and to be of low reliability, these problems can be solved by means of a dedicated maintenance system "TAKO" and its efficient testing environment.

Assembly: The assembly was time-consuming and mistakable task. However, we easily assembled it by taking the following measures:

- Coloring FRCs in accordance with their logical locations prevented misconnections.
- After positioning the FRCs in an open space, we moved them into the rack and connected with the boards.

Moreover, using the TAKO we found and corrected some errors quickly.

Figure 6 and 6 show the appearance of the INhardare.

Testing and Maintaining

In this section, we describe the way to test and to debug the IN.

Each switch node can be tested separately. However, it cannot be tested with an ordinary logic analyzer by the following reasons:

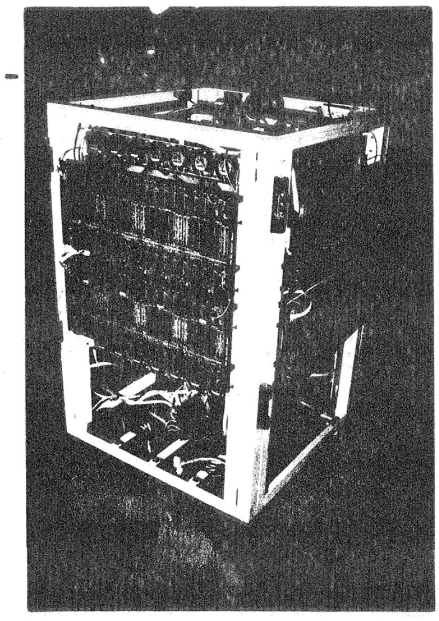

Figure 6: The IN hardware (finished assembling)

- Many pins of the switch must be monitored simultaneously.
- Data lines, which are of 32-bit width, are bidirectional.
- The switch must be tested by supplying with 10MHz clock pulse, which is the same as it operates actually.

For this purpose, we developed a dedicated maintenance system "TAKO", which comprises a host interface, a clock generator, and network analyzer. The network analyzer has eight probes and are useful for testing the switch node, because the switch node has eight ports (four inputs and four outputs).

TAKO has two features:

- It can emulate the function of communication controller in the processor in order to drive the IN with 10MHz clock.
- It can monitor data flowing in the IN.

TAKO provides several functions: testing the SUs, the switch nodes, the FRCs, the IN boards, and the IN and monitoring the IN.

We can operate the TAKO in an interactive environment on the X window system.

The TAKO and its interactive operational environment resulted in testing the IN efficiently

Using the TAKO, we can test the IN as easily as to deal with software, owing to controlling the system clock and the scan paths in the SUs.

Evaluation of the Electrical Characteristics

In this section, as the first step of the evaluation of the IN, we examined its basic parameters that were measured on the actual hardware.

First, we checked on the IN, as follows:

1. On all the switch nodes
2. On all the FRCs' physical connections and their abilities of transfer

Because each switch node on the IN boards is independent of other as mentioned in the section four, we were able to test the IN boards alone as to the item 1 above briefly and systematically. In consequence of examination into the topology, the item 2 can be achieved through the check on only 384 connection patterns out of 4096, which is the number of all the possible connections. Two steps above have proven correctness of the entire IN without the exhaustive test.

Next, we measured the basic performance of the IN. The followings were the test items and their results.

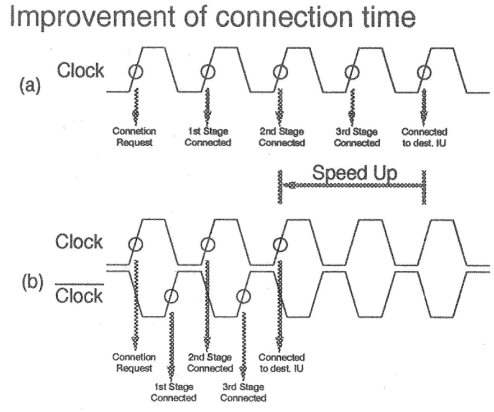

Figure 7: Speedup of connecting cycles using two phase clock

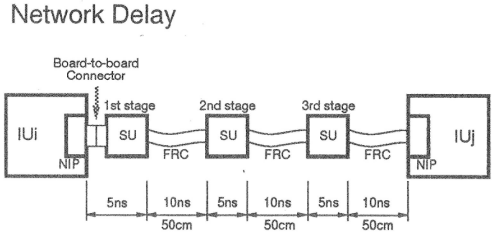

Figure 8: Delay time in transferring data in the IN

Establishing Connections: The overhead in establishing connections is one of shortcomings of the circuit switching. The 64 PEs operate synchronizing with a global clock. While data can pass through the switch node irrelevantly to the clock, the switch node alters its internal state synchronously to the clock. Thus, it takes three clocks to establish a connection, and the destination knows the connection after four clocks (fig. 7(a)). If switch nodes in the first and the third stage operate with the clock that is 180-degree fast in phase, it takes two clocks to establish a connection (fig. 7(b)).

Data Transfer: Circuit switching non-buffering network provides PEs direct communication channels like hardwired. It transfers data as electrical signal in the electrical conduction delay alone. According to the measurement, the delay in the SU and the FRC is 15 *ns* and 5 *ns*, respectively. Thus, the total delay from end to end is 60 *ns*. With the delay time against the global clock (10 MHz), synchronous transfer can apply.

Turning Over the Transfer Direction: Direction of data transfer is controlled with a state of the DIR control line, whose voltage level represents the direction. The IN transmits transition of the state to the destination in a delay of 60 *ns*. Each switch node turns the direction over according to the transition without synchronization with the global clock. Thus, turning over is finished within one clock cycle. We confirmed that data can be transferred bidirectionally with one clock cycle to turn the direction over.

Release Connections: Connection release requests are also transmitted in a delay of 60 *ns*. Each switch node receiving the release request accepts it at next rising edge of the clock, and releases the connection at the rising edge of next one. This sequence ensures the release operation without the global clock. Tests in various situations indicate its effectiveness.

Characteristics of the Electrical Transfer Lines: The SUs that the switch node comprises are CMOS devices. Therefore, a CMOS driver drive a CMOS input through a FRC. Waveform in the transmission-lines is desirable. We presume that it is due to the following reasons:

- Because the idea for assembly of the IN shortened FRCs and all drivers for FRCs are CMOS, the rising edge is dull as to exclude the reflection.
- The impedance of transmission lines is stable because every other line in the FRC was a grounded shielding line and one driver only drives one transmission line associated with one input pad.

Table 1: Performances of the IN

	Performance
Transfer Rate	40MBytes/s
Bandwidth	5GBytes/s (Two Networks' Total)
Connection Establishment	2 Clock Cycles (Minimum)
Transmission Delay	60 ns
Direction Turning-over	60 ns
Connection Release	1 Clock Cycle

In order to measure the rate of failure in transfer, we continued to establish a connection, to transfer some amount of data, and to release the connection for 24 hours. However, no error was observed.

Moreover, in order to confirm the operation in the heaviest situation, we tested the IN as follows. By means of multicasing (one-to-many connection) facility of the SU, make most of FRC (149 cables out of 192) be in use, and give all of them the same data changing like "010101...". In these state, we continued to establish a connection, to transfer data, and then release the connection. As a result, no error was observed.

Table 1 summarizes the measurement of the IN. This shows the IN is suitable for PIE64. While the paper examines basic parameters of the IN, it is also important to evaluate its performance in the situation that 64 PEs operate parallelly. Another paper will report the evaluation when the PEs are accomplished.

Conclusion

An interconnection network of PIE64 has several features: (1) circuit switching, (2) multistage network, (3) non-buffering in any node, (4) duplicated, and (5) supporting dynamic load balancing among PEs. First, this paper has discussed these features and showed that they were suitable for the interconnection network. Second, the physical implementation method of IN, and the process of its assembly has been reported. Finally, the way for testing, debugging, and maintaining the network has also been explaining. For all reasons, we conclude that the INthus made is able to meet the basic requirements for the use in PIE64.

Acknowledgements

We thank the other members of the team SIGIE, namely Tadashi Saito, Minoru Yoshida, Takeshi Shimizu, Kentaro Shimada, Takeshi Shimoyama, Yasuo Hidaka, Junichi Tatemura, Hidemoto Nakada, Atsuo Shouno, Takashi Matsumoto, and Hiroshi Miyaki.

This work is supported by Grant-in-Aid for Specially Promoted Research of the Ministry of Education, Science and Culture (No.62065002).

References

[1] Koike, H. and Tanaka, H.: "Parallel Inference Engine PIE64," in *Parallel Computer Architecture*, bit, Vol.21, No.4, 1989, pp. 488-497 (in Japanese).

[2] Koike H., Takahashi E., Yamauchi T. and Tanaka H.: "The High Performance Interconnection Network of Parallel Inference Machine PIE64," Computer Architecture Symposium, Information Processing Society of Japan, 1988.

[3] Sakai, S., Koike, H., Tanaka, H. and Motooka, T.: "Interconnection Network with Dynamic Load Balancing Facility," Computer Architecture Symposium, Information Processing Society of Japan, 1988.

[4] Hidaka, Y., Koike, H., and Tanaka, H.: "The Architecture of the Inference Unit of Parallel Inference Engine PIE64," CPSY 90-44, SWoPP '90, Institute of Electronics, Information and Communication Engineers, July, 1990 (in Japanese).

[5] Takahashi, E., Koike, H., and Tanaka, H.: "Implementation and Evaluation of the Interconnection Network of PIE64," Proc. of JSPP'90, pp.89-96, Information Processing Society of Japan, May 1990 (in Japanese).

[6] L.N.Bhunyan, "Interconnection Network for Parallel and Distributed Processing," Computer, June 1987, pp.9 - 12.